Tag Anti-collision Algorithms in RFID Systems - a New Trend.

MAJID ALOTAIBI, ADAM POSTULA, and MARIUS PORTMANN
Information Technology and Electrical Engineering
The University of Queensland
Brisbane QLD 4072
AUSTRALIA
majid@itee.uq.edu.au

Abstract: - RFID is a wireless communication technology that provides automatic identification or tracking and data collection from any tagged object. Due to the shared communication channel between the reader and the tags during the identification process in RFID systems, many tags may communicate with the reader at the same time, which causes collisions. The problem of tag collision has to be addressed to have fast multiple tag identification process. There are two main approaches to the tag collision problem: ALOHA based algorithms and tree based algorithms. Although these methods reduce the collision and solve the problem to some extent, they are not fast and efficient enough in real applications. A new trend emerged recently which takes the advantages of both ALOHA and tree based approaches. This paper describes the process and performance of the tag anti-collision algorithms of the tree-ALOHA trend.

Key-Words: RFID, Collision, Frame, Algorithm, Aloha, Tree, Tag

1 Introduction

Radio frequency identification (RFID) is a contactless, without line-of-sight, low-power and low-cost wireless communication technology that provides automatic identification and data collection. In addition, RFID can read multiple tags simultaneously and work in harsh environments [2].

Although RFID technology is already widely applied in manufacture, supply chain, retail inventory control, transportation, healthcare, agriculture, construction, etc [3, 4] it still needs substantial improvements. Combining RFID with the Internet, database and middleware can change everyday objects into mobile network nodes, and then it becomes easy to track, trace, monitor, and perform some actions on those tagged objects. This will change a lot of different applications fields and create huge new opportunities for designing smart automated systems that can enhance human productivity and efficiency.

The RFID system contains number of tags with unique serial number, a reader to communicate with tags and collect information from tags, and application processor (see fig.1). A RFID reader identifies an object through RF wireless communications with the tag which has a unique ID and information, and is attached to the object. The tag sends its own ID to the reader for tag identification. The reader should be able to recognize tags as quickly as possible [5]. However, the signals in both directions between the reader

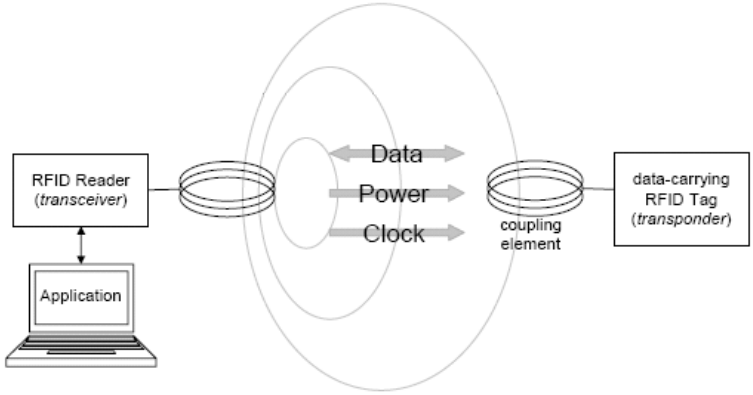


Fig.1: The main components of RFID system [1].

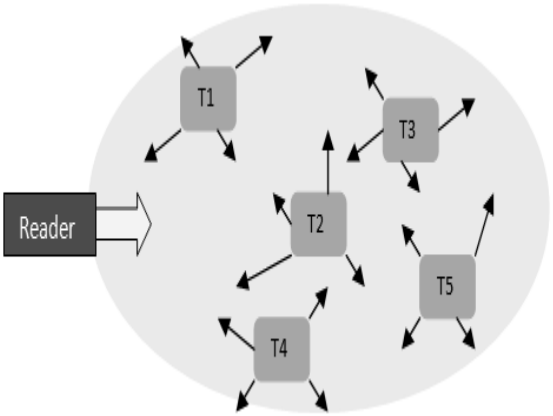


Fig.2: The cause of collisions.

and the tag may collide because readers and tags communicate over a shared wireless channel (see fig.2). As a consequence, either the reader may not identify all tags or a tag identification process may suffer from long delay. Thus, anti-collision algorithms which provide quick and correct recognition regardless of the occurrence of collisions are required [6].

Tag anti-collision algorithms can be classified into two broad groups: probabilistic algorithms and deterministic algorithms. Probabilistic algorithms, such as ALOHA, slotted ALOHA and frame slotted ALOHA, decrease the occurrence possibility of tag collisions by allowing the tags to transmit their own serial numbers at a distinct time. In ALOHA, tags randomly choose their transmission time and, in slotted ALOHA, tags transmit only at the beginning of a timeslot which is a certain time period. Frame slotted ALOHA which is the best choice of probabilistic algorithms configures a frame with many timeslots [5, 7-10]. As a tag sends its ID only at a single timeslot in every frame, frame slotted ALOHA decreases the collisions. However, probabilistic algorithms cannot completely avoid collisions. Moreover, these algorithms cannot guarantee that all tags are recognized within a certain time period if the number of tags is not known in advance. Therefore, a specific tag may not be identified for a long time, and cause what is called tag starvation problem [11].

For deterministic algorithms, the reader sends a request command, which is based on the tags IDs, to tags in the interrogation zone to respond. These methods are based on tree anti-collision algorithms such as the binary tree and the query tree algorithms. In the binary tree algorithms, if a collision occurs, the reader splits a set of the colliding tags into two groups until a tag is identified without collision. The binary tree algorithms have many techniques, for instance, polling a list of tags' IDs, or performing some variations of binary search algorithm [11-13]. In the query tree algorithms, a reader sends out a prefix in each communication iteration and tags simply respond with their IDs if the prefix matches parts of their IDs. If a collision occurs for a particular prefix, the reader ignores the response and expands the prefix by one bit. The efficiency of the identification process for deterministic algorithms can be affected by the length and distribution of tag IDs, and the tag population size [14].

There is much published research on improving probabilistic and deterministic algorithms in different ways. Some works try to improve the way of choosing an optimal frame size by finding an appropriate method of estimating the number of tags in the interrogation zone [15-17]. Others come with algorithm in which the number of responding tags is restricted in case of high number of present tags in the reading area [18-20]. In addition, some researchers modify the binary tree algorithms by reducing the long identification delay [12, 13, 21-23]. Although these studies reduce the collision and solve the problem to some extent, they are not fast and efficient enough in real life applications. Since the tree based RFID tag anti-collision algorithms achieve full read rate, and the slotted ALOHA based tag anti-collision algorithms provide simple implementation and good performance for small amount of tags, some researchers focus on how to take the advantages of both algorithms [24-27]. Thus, a new trend of tag anti-collision algorithms emerged which is tree-ALOHA algorithms. The tree-ALOHA algorithms can be considered as deterministic algorithms with improved performance due to ALOHA process.

In the second section of this paper, the ALOHA based anti- collision algorithms are shortly discussed, the third section explains the tree based algorithms. The new developments of tree-ALOHA anti-collision algorithms are described and analyzed in the fourth section, and the conclusions are drawn in the last.

2 ALOHA based anti-collision algorithms

All probabilistic algorithms are basically modified forms of ALOHA algorithm. Fig.3 shows a variety of probabilistic algorithms. These algorithms

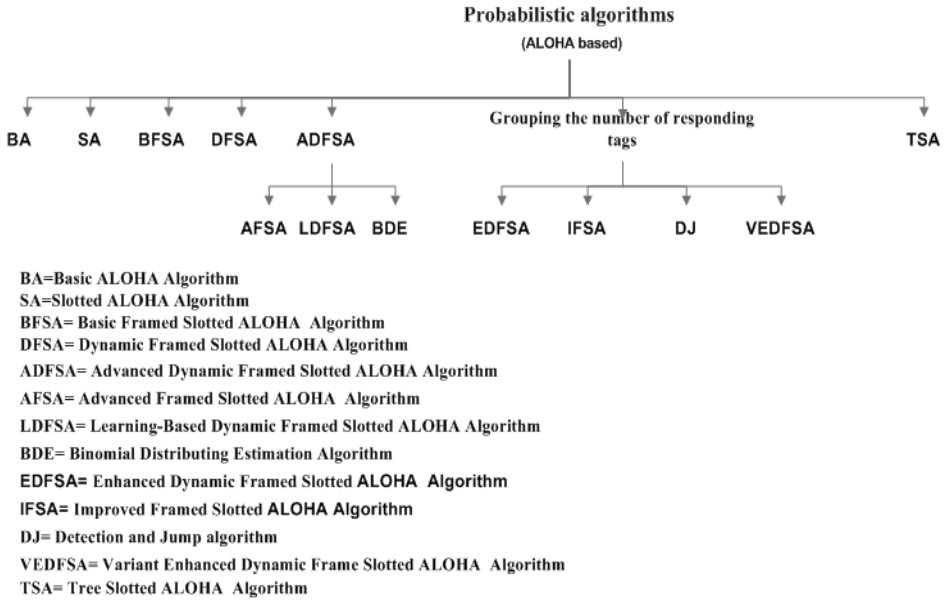


Fig.3: some existing algorithms of probabilistic algorithms

decrease the probability of the occurrence of tag collisions since tags try to response in distinct times.

2.1 Basic ALOHA and Slotted ALOHA Algorithms

The simplest of all the anti-collision algorithms is ALOHA algorithm. The reader just sends a request command to the tags which are present in the interrogation zone of the reader. Then, each tag transmits its ID number to the reader in randomly selected time. Therefore, there is a certain probability that two tags may transmit their ID numbers at the same time and the data will collide with one another. However, if just one tag sends its ID number in a certain time, this results in successful identification [28]. The efficiency of this algorithm will drop down if there is a large number of present tags in the field of the reader or if the tags have huge amount of data [28].

One possibility for optimizing the performance of the ALOHA algorithm is the slotted ALOHA algorithm. In this protocol, tags may only start to transmit data at defined, synchronous points in time (slots). The synchronization of all tags must be controlled by the reader. Thus, this can be called a reader-driven TDMA anti-collision protocol [29]. The S-ALOHA algorithm has better performance than the basic ALOHA because of the synchronization between the reader and the tags. Moreover, the defined points of time that tags start response in them reduce the probability of collision. However, the S-ALOHA algorithm still has the weak point that it provides poor performance when the number of tags increases [12, 29].

2.2 Basic Framed Slotted ALOHA (BFSA) Algorithm

A frame is a time period between requests of a reader and also contains a number of slots. In BFSA algorithm, the frame size is fixed and not changed during the identification process. The reader broadcasts the request command which includes the frame size and the random number. Each tag uses the received random number to select a slot in a frame, and respond in this slot [30].

The process of BFSA algorithm can be shown in Fig.4. The frame size in this example is three slots. In the first round, Tag 1 and Tag 5 simultaneously send their ID number in slot 1. Tag 2 and Tag 3 respond in slot 3. Because Tag 1, 2, 3 and 5 cause collisions, they must respond in the next read cycle. Tag 4 can be identified in the first round because Tag4 just responses in the time slot 3 (see fig.4). In the second round, Tag 1 and Tag 3 are identified respectively in slot 2 and 3. However, Tag 2 and Tag 5 collide each other in slot 1. Therefore, they must respond in the next round.

Although the fixed frame size of BFSA algorithm provides simple implementation, its disadvantage is that the efficiency of tag identification is decreased. For example, if there are a large number of tags in the field of the reader,

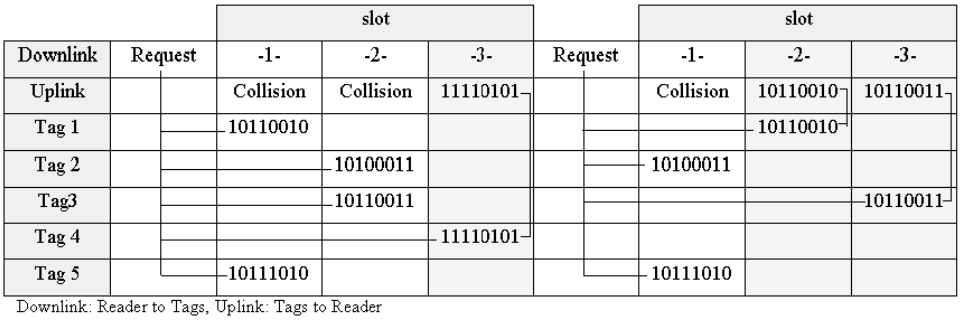


Fig.4: The process of BFSA algorithm

no tag might be recognized through the read cycles and all slots may be filled with collision. Also, there is a waste of time if a large frame size is used in the case of small number of tags [30].

2.3 Dynamic Framed Slotted ALOHA (DFSA) Algorithm

In DFSA algorithm, the frame size is changed for efficient tag identification. The frame size can be determined by using the information such as the number of slots used to identify the tag, the number of collided slots and the number of empty slot. Thus, DFSA algorithm can improve the BFSA algorithm that is inefficient to identify the tag [12, 31, 32].

DFSA algorithm controls the frame size using the information of the previous read cycle, for example, the number of empty slots, collided slots and successful slots (slots filled with one tag). If the collision probability is over the upper threshold, the reader increases the frame size. However, if the collision probability is smaller than the lower threshold, the reader decreases the frame size. The reader begins a read cycle with the minimum frame size, therefore, when the number of tags is small, it can recognize the tags efficiently without raising the frame size much. In the case of large number of tags, the reader changes the frame size to decrease the collision probability [12].

DFSA algorithm is more efficient than BFSA algorithm because the reader regulates the frame size based on the number of tags. However, changing the frame size alone cannot decrease the tag collision in case of very large number of tags because the frame size cannot be increased indefinitely.

2.4 Advanced Dynamic framed Slotted ALOHA Algorithms

AFSA algorithm regulates the frame size by estimating the number of tags. Thus, it has better performance than BFSA algorithm. There are many existing estimation functions.

Vogt [16] proposed two methods to estimate the number of tags around the reader. The first estimation method is obtained through the observation that a collision involves at least two different tags. Therefore a lower bound (LB) on the value of the estimated number of tags can be obtained by the simple estimation function as in equation (1) [16].

$$E = Number\ of\ collided\ slots\ X\ 2 \quad (1)$$

The second method is called CIILB (Chebyshev Inequality Improved Lower Bound) and can be derived as follows. Chebyshev's inequality tells us that the outcome of a random experiment involving a random variable **X** is most likely somewhere near the expected value of **X**. Thus, an alternative estimation function uses the distance between the read results, which are frame size, number of successful slot, number of collided slot, and number of idle slot and the expected value vector to determine the number of tags for which the distance becomes minimal as in equation (2) [16].

$$\epsilon_{vd}(N, c_0, c_1, c_k) = min \left| \begin{pmatrix} a_0^{N,n} \\ a_1^{N,n} \\ a_{\geq 2}^{N,n} \end{pmatrix} - \begin{pmatrix} c_0 \\ c_1 \\ c_k \end{pmatrix} \right| \quad (2)$$

Where N and n respectively denote the frame size and the number of tags. $a_0^{N.n}$, $a_1^{N,n}$, $a_k^{N,n}$ are respectively the expected number of the empty slots, slots filled with one tag, and slots with collision.

The third method is Binomial Distributing Estimation (BDE) function. Expectation value of collision percent in *N* slots is [17]:

$$E(C) = 1 - \left(1 - \frac{1}{N}\right)^n \left(1 + \frac{n}{N-1}\right) \quad (3)$$

After the end of an identifying round, actual collision percent can be gained:

$$C = \frac{collision\ slot\ number}{N} \quad (4)$$

Then,

$$C = 1 - \left(1 - \frac{1}{N}\right)^n \left(1 + \frac{n}{N-1}\right) \quad (5)$$

We can estimate tag number *n* through (5) after each identifying round.

The fourth method is using the maximum throughput condition to estimate the number of tags in the vicinity (see equation (6)). Let *Mcoll* be the number of collided slots in a frame after a round [8].

Number of estimated tags = 2.3922 × *Mcoll.* (6)

ADFSA algorithm has the same problem as DFSA algorithm that the frame size cannot be increased indefinitely as the number of tags increases. Therefore, this algorithm gives good performance when the number of tags is small, however, it shows poor performance when the number of tags becomes large [33].

2.5. Grouping the number of responding tags

In the other framed slotted ALOHA algorithms, there is a problem that the frame size cannot be increased indefinitely, in order to maximize the system efficiency. However, this problem can be addressed by restricting the number of responding tags approximately the same as the frame size. For example, Enhanced Dynamic Framed Slotted ALOHA (EDFSA) algorithm can solve this problem by groping the unread tags using modulo operation, Equation (7) [18].

$$M(The\ number\ of\ groups) = \frac{The\ number\ of\ unresd\ tags}{N(maximum\ frame\ size)} \quad (7)$$

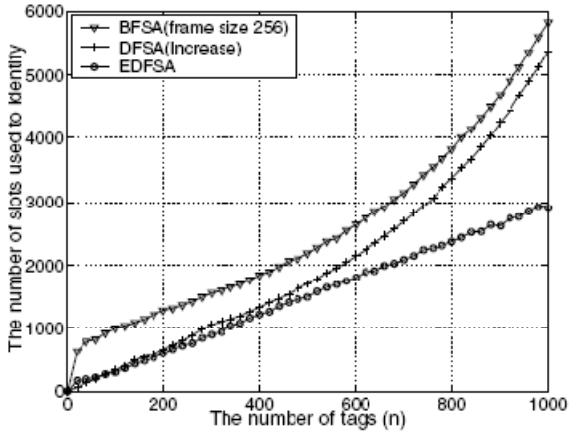


Fig.5: The total number of slots used to identify up to 1000 tags [18]

EDFSA algorithm shows higher performance of 100% and 85% than BFSA and DFSA algorithms respectively see fig.5 [18].

The performance of Variant Enhanced Dynamic Frame Slotted ALOHA (VEDFSA) Algorithm is better than the EDFSA algorithm by promoting the group solution of the algorithm. The group in the EDFSA algorithm is fixed, and is not changed during the whole identification process. However, in VEDFSA algorithm, the group will be changed dynamically during the whole identification process. The tags are divided into several groups aim to fulfill optimal efficiency in each group in the first round. After the first round of reading, each group has decreased a certain number of identified tags. In the second round, the unread tags are divided into new groups; each group is expected to fulfill the higher performance. The tags are identified in the end of this process [34].

Improved Framed Slotted ALOHA (IFSA) Algorithm has the same concept as EDFSA algorithm which restricts the number of responding tags to achieve high identification performance but with different treatment [20].

The IFSA method limits the number of responding tags by comparing a part of the identification number (ID) saved in the tag with the comparison bit. The reader sends the comparison bit, the comparable position of ID in the tag and the comparison bit length to tags present in the communication range. Then, when tags receive this information, they compare the comparison bit with a part of ID. Thus, if the part of ID is smaller than the comparison bit, the tags responds to the reader [20].

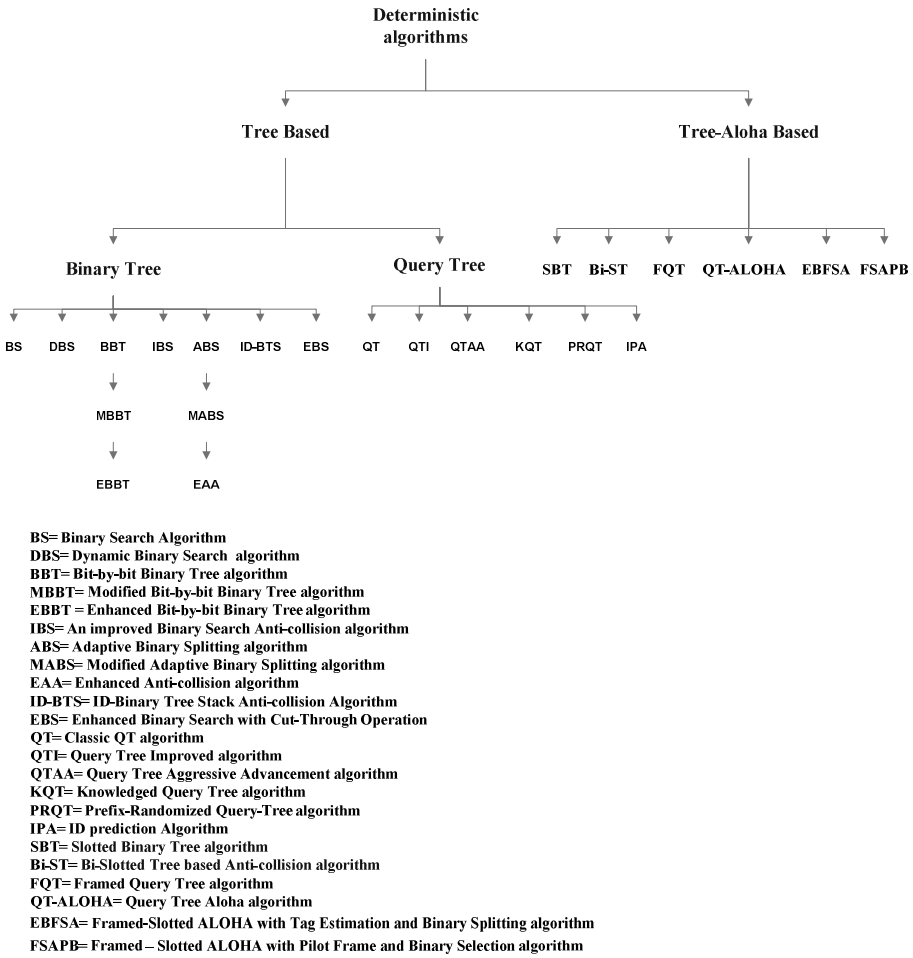


Fig.6: Tree based (deterministic) algorithms

2.6 Tree Slotted ALOHA (TSA) algorithm

TSA is a modified version of slotted ALOHA protocol. In this method, the collision is resolved as soon as it happens. Therefore, if the collision is detected in a slot, the reader requests only tags which collided in that slot in the next read cycle. Two tags do not collide in a frame, might collide in the next frame in the other framed slotted ALOHA protocols. However, in TSA method, the previous situation is avoided because when a collision is produced in a slot, only the tags that cause this collision are requested in the next round. This results in enhanced performance [19].

3 Tree based algorithms

Most Tree based algorithms are basically modified forms of Binary or Query Tree algorithms. Fig.6 shows variety of Tree based algorithms.

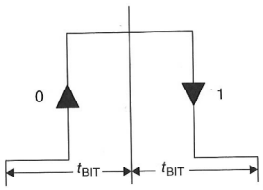


Fig.7: Bit coding using Manchester code [12].

3.1 Binary Tree Algorithms

The collision can be solved in Binary Search (BS) algorithm by gradually decreasing collided bits in tag's serial number. The reader can detect the position of collided bit by using a suitable bit coding; for example, the Manchester code makes it possible to trace a collision to an individual bit. In this bit coding, the value of the bit can be determined by the way of transition. If the transition is positive, that means the logic is '0'. However, the negative transition means the logic is

'1' (see fig.7). The "no transition" means an error occurs [12].

If more than one tag simultaneously transmits bits of different values, the positive transitions cancel the negative transitions of the received bits. No transition, therefore, is detected and that leads to an error. Hence, the collision in individual bit can be traced (see fig.8) [12].

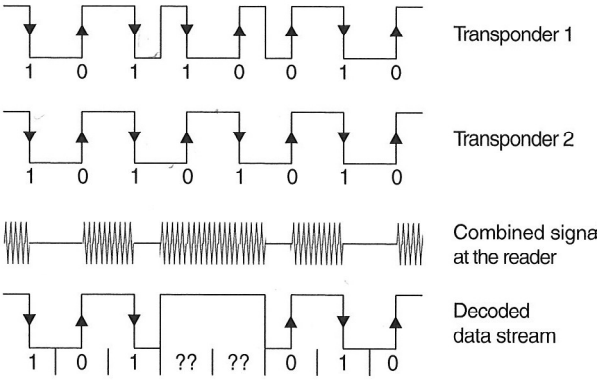


Fig.8: Collision behavior for Manchester code [12].

BS algorithm divides the responding tags to two groups. It allows the tags that have first collided bit '0' to respond to the next request, however, tags with first collided bit '1' do not respond. Assuming there are four tags in the interrogation zone of the reader. At the beginning, the reader sends a request asking tags that have serial numbers less or equal to 11111111b to respond. Because 111111111b is the highest serial number, all tags will respond to the reader in the first iteration by sending their own serial numbers. A collision is sensed in the received serial number at bit 0, bit 4 and bit 6. Therefore, there are eight (2^3=8) or less tags in the reading range. Bit 6 is the highest collided bit. This means there is at least one tag has the serial number less than 10111111b. Thus, in order to limit the number of responding tags, the reader sends in the next iteration a new request for tags that meet the condition of (≤10111111b). In this example, the received serial number will have collisions at bit 0 and bit 4. Hence, the procedures will be repeated. The reader will request tags that have the serial numbers less than or equal 10101111b. Tag 2 is only located in this iteration (3rd iteration). The reader then selects Tag 2 and starts the communication. By repeating these procedures, all tags can be identified by the reader (see fig.9) [12].

The Adaptive Query Splitting algorithm which is an improvement of the Query Tree algorithm, and the Adaptive Binary Splitting algorithm which is based on the Binary Tree algorithm are proposed by Myung [11, 14, 35]. The proposed algorithms use information obtained from the last process of tag identification in order to reduce the collision.

3.2 Query Tree (QT) Algorithms

In QT algorithm, the reader broadcasts a prefix and tags in the vicinity which have the matching ID to the prefix, response. If a collision occurs, the reader asks for one bit longer prefix until no collision is detected. The round is terminated by the reader and a new round of queries is started with another prefix, once a tag is identified (See Table 1) [11, 36]. Query Tree Improved (QTI) algorithm is a modified form of QT algorithm. QTI algorithm avoids the queries that indeed will yield collision. For example, assume that a query with prefix "q" produces a collision, and the query with prefix "q0" produces an empty slot. Then, the reader neglects the query "q1" and does only the queries "q10" and "q11" [36].

Downlink (Reader-Tag)	Request <11111111	1st iteration	Request <10111111	2nd iteration	Request <10101111
Uplink		1X1X001X		101X001X	
Tag 1		10110010		10110010	
Tag 2		10100011		10100011	
Tag 3		10110011		10110011	
Tag 4		11100011			

Downlink (Reader-Tag)	Request <10101111	3rd iteration	Select 10100011	Read/Write
Uplink		10100011		
Tag 1				
Tag 2		10100011		10100011
Tag 3				
Tag 4				

Fig.9: The process of BS algorithm [12].

```
Reader procedure:
1. prefix = empty;
2. query(prefix,0);
3. query(prefix,1);

query(char[] prefix,char c):
4. prefix+ = c;
5. broadcast(prefix);
6. receiveAnswers;
7. if (channelStatus= 1) then
       sendAck;
8. else if (channelStatus> 1) then
9.         query(prefix,0);
10.        query(prefix,1);

Tag procedure:
11.   identified = false;
12.   while (not identified) do
13.       receive(prefix);
14.       offset = prefix.length;
15.       if (prefix = myID[0...offset − 1]) then
16.           send myID;
17.           if receivedAck then identified = true;
```

Table 1: Pseudo-code of QT protocol [36]

Query Tree Aggressive Advancement (QTAA) algorithm uses quaternary tree not binary tree. For example, suppose that a query with prefix 'q' results in a collision, the reader performs the four queries 'q00', 'q01', 'q10' and 'q11' [36].

Query tree algorithms have the advantage to be memoryless because tags do not need additional memory except to store the ID. Therefore, they require low functional and less expensive tags. However, because they use prefixes, the performance of these algorithms is sensitive to the distribution of tag IDs.

Prefix-Randomized Query-Tree (PRQT) algorithm differs from the normal Query-Tree (QT) algorithm in using randomly chosen prefixes by tags rather than using their ID-based prefixes as in QT. Thus, the length and distribution of tag's ID do not affect the identification time of PRQT anymore [37].

4 Tree – ALOHA based algorithms

Performance in tags collision problem is usually measured as system efficiency which is basically the ratio between the number of tags to be identified, and the number of queries or time slots used in the whole identification process. All the algorithms reviewed in the previous sections show an average performance well below 50%. In fact, the best performing algorithms, namely QT and TSA, achieve around 40% of system capacity on average, in their best performance [19, 36].

The low performance of the ALOHA algorithm is caused by the tag starvation problem presented before. Long identification time in the tree based anti-collision algorithms is inherent to the method and shows especially with large number of tags with long IDs. On the other hand, the tree based RFID tag anti-collision algorithms achieve full read rate, and the ALOHA based tag anti-collision algorithms provide simple implementation and good performance for small amount of tags. Thus, combining both methods and taking their advantages can lead to better performance.

This section presents some of the algorithms which combine different algorithms from the two methods.

4.1 Slotted Binary Tree (SBT) algorithm

This algorithm treats the collision as soon as it occurs. When a collision occurs in slot i, all tags, which do not respond in the collided slot i, wait until the collision is addressed. Then, the reader asks the tags involved in the collision to choose randomly group '0' or group '1'. The tags that selected group '0' respond in slot i+1 while the tags in the other group wait until all tags in group '0' are successfully identified. If slot i+1 is either idle or successful slot, tags of group '1' respond in slot i+2. Otherwise, there is a collision; the same procedures are repeated to solve the new collision [38].

If there are n tags in the interrogation zone of the reader, the iteration of SBT algorithm (I_{SBT}) can be calculated by the following formula [38]:

$$I_{SBT} = 1 + \sum_{k=2}^{n} \binom{n}{k} \frac{2(k-1)(-1)^k}{[1-p^k-(1-p)^k]} \qquad (8)$$

4.2 Bi-Slotted Tree based Anti-collision algorithm

Bi-slotted Tree based algorithms which will be described in this section are bi-slotted query tree algorithm (BSQTA) and bi-slotted collision tracking tree algorithm (BSCTTA). The processes of both algorithms (BSQTA, BSCTTA) can be explained in fig.10 and fig.11, respectively [21, 24].

The main aspect of these algorithms is that they send two 'n length query bits', which have the same n-1 bits and the different last bit. The following steps describe the identification process of these two algorithms [21, 24]:

1- Request: The reader broadcasts n-1 length prefix.
2- Grouping: Tags in the interrogation zone of the reader respond if the prefix matches the first n-1 bits of their IDs. These tags

respond in one of two time slots depending on the n_{th} bit. If n_{th} bit is '0', tags select the first slot; otherwise, if n_{th} bit is '1', tags select the second slot. Hence, the selected slot shows the value of n_{th} bit.

- BSQTA: tags transmit their IDs from n+1 bit to the end bit.
- BSCTTA: tags transmit their IDs from n+1 bit to the time that they receive ACK signal, which says there is a collision, from the reader.

3- Decision:
- If a collision occurs, the reader stores a new prefix at (LIFO).
  i. BSQTA: the connection of n-1 length prefix and the illustration of the selected slot.
  ii. BSCTTA: the connection of n-1 length prefix and the illustration of the selected slot, and the bits received before the collision happen.
- If the last bit is collided, the reader assumes there are two tags.
- If there is no collision, the tag is identified by the reader.

4- The steps are repeated until the LIFO is empty.

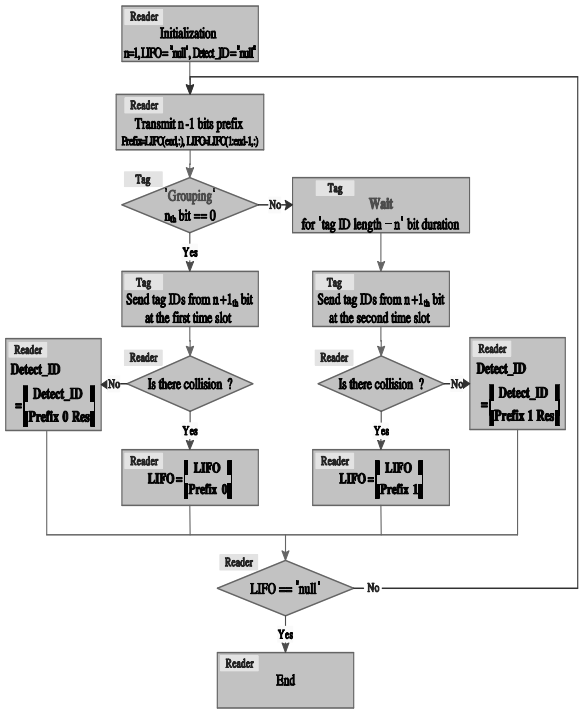


Fig.10: The Bi-Slotted Query Tree Algorithm (BSQTA)[24]

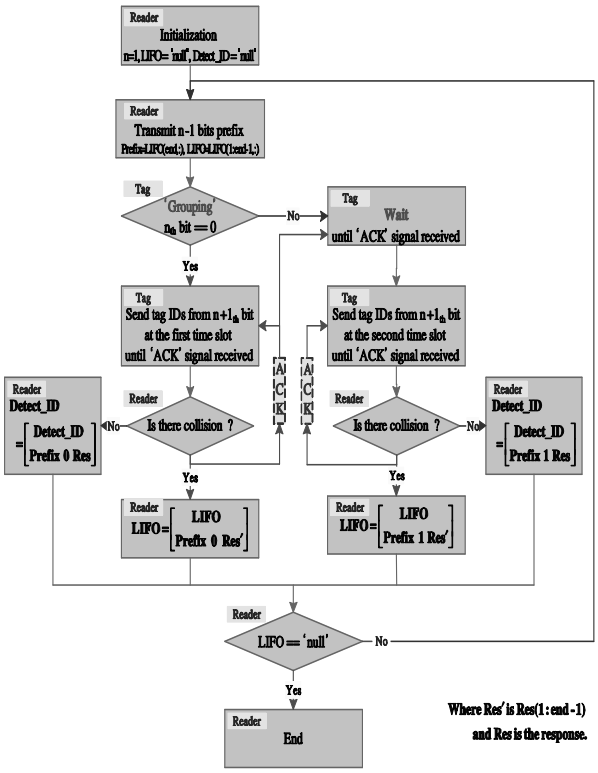
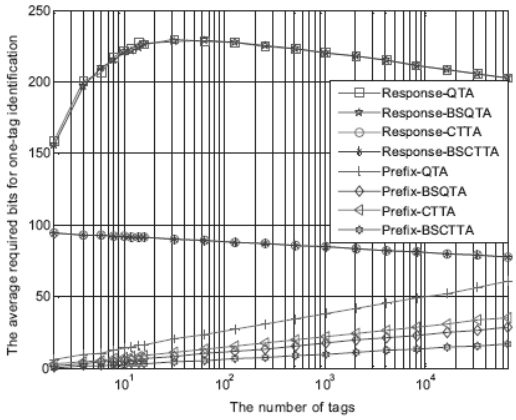


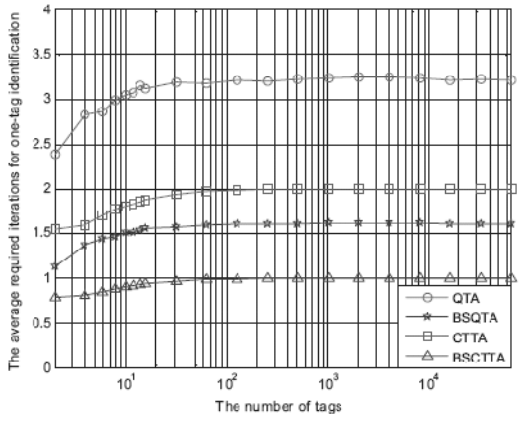
Fig.11: The Bi-Slotted Collision Tracking Tree Algorithm (BSCTTA [21, 24]).

By using BSQTA and BSCTTA, the average of required prefix overhead is reduced to the half of the prefix overhead in the tree based algorithms, QTA and CTTA. Fig.12 shows better performance of BSQTA and BSCTTA than other tree based anti-collision algorithms [21, 24].

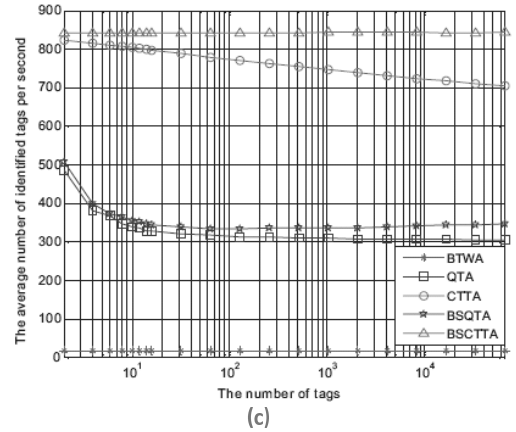
The average required prefix and response bits for one tag identification using different anti-collision algorithms are shown in Fig.12-a. In the comparison of BSQTA and QTA, the average required prefix in BSQTA is decreased to approximately 50% of the prefix required in QTA; however, there is no change in the average required response bits for one tag identification. Therefore, less bits are required in BSQTA for one tag identification than QTA. Furthermore, the differences in performance between BSQTA and QTA in the number of required bits to identify one tag increases as the number of tags increases (see Fig.12-a). To also evaluate the system performance by using another mean, the iteration number to identify one tag is measured. As seen in fig.12-b, BSQTA requires half the number of iterations in QTA. Eventually, BSQTA performs better than



(a)



(b)



(c)

Fig.12 (a) The average required prefix and response bits for one tag identification. (b) The average required iterations for one tag identification. (c) The average number of identified tags per second. [24].

QTA in both the average required bits and the average required iterations for one tag identification [24].

The same pattern is in the comparison of BSCTTA and CTTA. BSCTTA achieves better performance than CTTA in the required bits to identify one tag. The average required prefix in BSCTTA is decreased to approximately half the prefix required in CTTA; however, there is no change in the average required response bits for one tag identification (see fig.12-a). Therefore, less bits are required in BSCTTA for one tag identification than CTTA. Moreover, as shown in fig.12-b, BSCTTA requires half the number of iterations in CTTA [24].

Finally, the average number of successful identified tags per second using different algorithms is shown in fig.12-c. Obviously, BSQTA and BSCTTA perform faster in identification process than QTA and CTTA respectively [24].

4.3 Framed Query Tree (FQT) algorithm

In this algorithm, tags randomly divided into frame units [27]. QT algorithm is used to identify tags in each unit. The process of FQT algorithm is describes as follow:

In the beginning of the identification process, the reader broadcasts a request to all tags in the interrogation zone asking for their ID. This request includes an epoch size which is the number of the total frames (see fig.13). Then each tag randomly chooses a frame and responds only when the reader queries its own frame. Within each frame, the reader uses QT algorithm to identify the tags of this frame. Thus, the reader sends tags the epoch size as well as the prefix of ID. If the selected frame is the same as sent one, the tag responds its ID matches the prefix (QT algorithm). Since all tags are identified in the frame, the reader continues to the next frame. The above process is repeated for every frame [27].

Fig.13 indicates an example of the identification process of FQT algorithm when the number of tags in the vicinity is 8 and the number of frame is 4 [27].

To improve the performance of FQT algorithm, an appropriate epoch size should be determined. QT algorithm gives the best performance when the tree depth does not exceed 2 (frame 3 in fig.13). When the number of tags to be identified is N and the epoch size is ES, the mist ideal ES can be determined as follow [27]:

$$N = 2 * ES \quad (9)$$

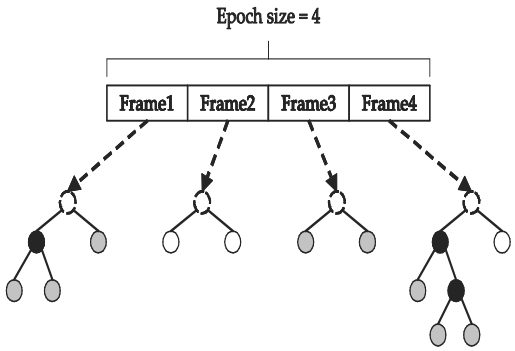


Fig.13: An example of identification process in FQT algorithm [27].

However, it is difficult to determine a proper epoch size in the beginning of the identification process because N is not known. Therefore, the final FQT algorithm uses FFT (First Frame Test). FFT starts the process from a small number of frames and if the collision in the first frame exceeds a collision threshold, FFT increases the epoch size. As mentioned that all tags are randomly divided in frames, if there is many collisions in the first frame, the other frames are more likely to have the same condition [27].

Frame 3 in fig.13 shows the best performance of the algorithm when the number of tags is 2 with the depth being 1. Hence a threshold is required in order to protect the tree from becoming deeper than this [27].

4.4 Query Tree Aloha (QT-ALOHA) algorithm

QT-ALOHA algorithm is a combination of Framed Slotted ALOHA (FS-ALOHA) algorithm and QT algorithm.

FQT algorithm basically builds FS-ALOHA algorithm and uses QT algorithm as actual tag identification process. Nevertheless, QT-ALOHA algorithm uses FS-ALOHA algorithm as actual tag identification process and QT algorithm as a big picture. The process of QT-ALOHA algorithm is describes as follow [27]:

In the beginning of the process, the reader sends a request that includes a prefix and a frame size together. Then, only tags with matching IDs are treated using FS-ALOHA algorithm and responds in the transmitted frame size. During FS-ALOHA algorithm, if a collision occurs even in a single slot, it is considered as a collision of QT algorithm. Thus, a new prefix is generated and added to the queue [27].

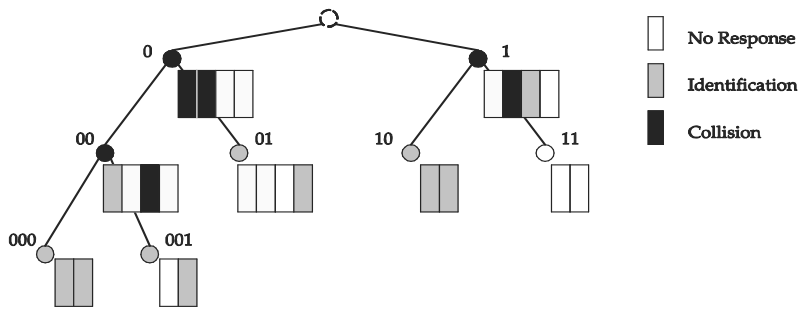
The example in fig.14 shows the process of QT-ALOHA algorithm. In this example, it is assumed that the number of tags to be identified is 8 and the initial frame size starts from 4. The table in fig.14 explains the rest of the process [27].

Fig.15 and fig.16 indicate that FQT algorithm in particular, has illustrated a big performance improvement [27].

Fig.15 shows how many queries-responses are needed to successfully identify one tag. Comparing that between algorithms, FQT shows best performance than any other algorithms which is around 10 to 50 percent of times than many other existing anti-collision algorithms [27].

4.5 Framed-Slotted ALOHA with Tag Estimation and Binary Splitting (EBFSA) algorithm

EBFSA has two phases: estimation phase and identification phase. The reader estimates the



Query	0	1	00	01	10	11	000	001
Frame Size	4	4	4	4	2	2	2	2
Response	collision	collision	collision	identified	identified	no res.	identified	identified

Fig.14: An example of identification process in QT-ALOHA algorithm [27].

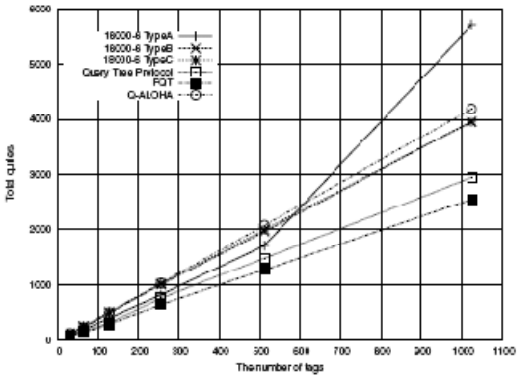


Fig.15: Comparison of query-response number [27]

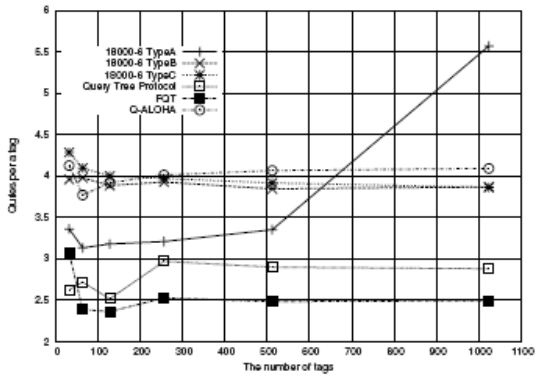


Fig16: Comparison of query-response number needed for identification per tag [27]

number of tags in the vicinity in the estimation phase and tags send their IDs during the frame interval in the identification phase. The estimation phase directs the tag estimation as Dynamic Framed Slotted ALOHA (DFSA) and uses the fixed frame size L_{est} (see fig.17). If the probability of collision is higher than a threshold P_{coll_th}, the frame size is decreased by a factor of f_d using the bit mask in a query frame. This process is repeated until the probability of collision P_{coll} becomes lower than the threshold P_{coll_th}. Then, the number of tags n can be estimated from P_{coll} and $L_c=L_{est}$ by using equation (10) [26].

$$P_{coll} = 1 - P_{idle} - P_{succ} = 1 - \left(1 - \frac{1}{L_c}\right)^n - n \cdot \frac{1}{L_c}\left(1 - \frac{1}{L_c}\right)^{n-1} \quad (10)$$

In this algorithm the tag estimation method can be computed regardless of an initial frame size and the actual number of tags to be identified. The actual estimation is preceded once which saves a lot of time. Then, the identification phase starts. Each tag chooses randomly a counter value. Then, the reader determines the optimal frame size L depending on the estimated number of tags in the estimation phase [26].

After that, every tag chooses randomly a counter value. In each timeslot, tag has different counter value. Tag decreases its counter by 1 for each timeslot. The tag sends its ID once its counter becomes zero. If there is a collision, the algorithm treats the collision by the binary selection. Thus,

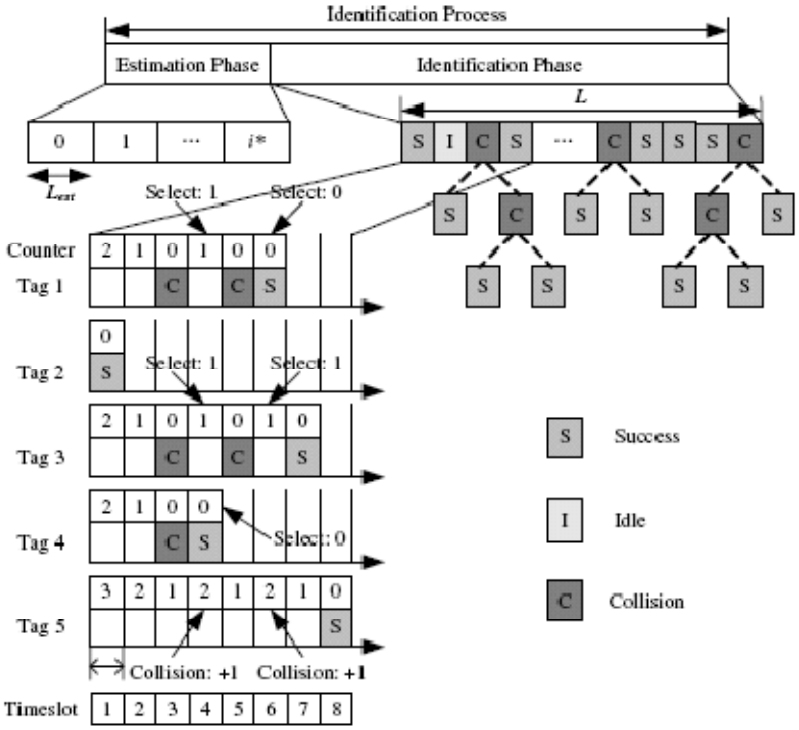


Fig.17: Identification process of our proposed algorithm and its example [26].

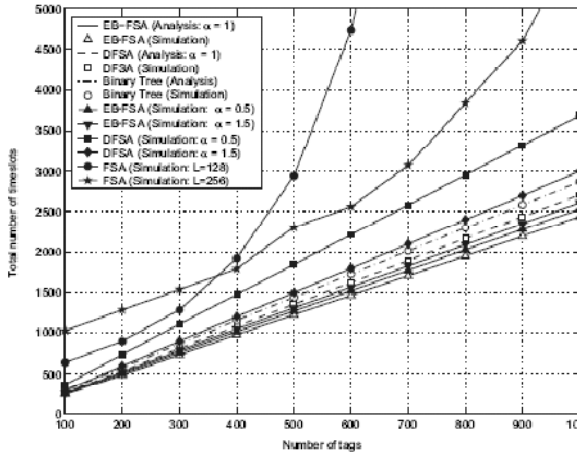


Fig.18: The total number of time slots used to identify tags with the varying the number of tags and the accuracy of tag estimation [26].

collided tags randomly select the value of their counters from 0 or 1; however other tags increase their counter by 1. For example, fig.17 shows that tag 1, 3 and 4 collide at timeslot 3 and to cope with this problem, they select 0 or 1 randomly, and tag 5 increases its counter by 1. In timeslot 4, tag 4 is successfully identified since its counter is zero. The same process continues until all tags are successfully identified [26].

In Dynamic Framed Slotted ALOHA (DFSA) algorithm, many frames are wasted because collided tags send their IDs in many frames in order to solve the collision. Nevertheless, in EB-FSA algorithm, the frame size is determined accurately in the estimation phase and collision is treated by binary splitting without creating more frames in the identification phase. Thus, the process of identification in EB-FSA algorithm is fast [26].

As shown in fig.18, EB-FSA uses 2433 slots which is smaller than DFSA (2709 slots) by 10.2% and the binary tree algorithm (2869 slots) by 15.2% [26].

4.6 Framed – Slotted ALOHA with Pilot Frame and Binary Selection (FSAPB) algorithm

In FSAPB algorithm, equation (10) is used to estimate the number is used for tag estimation. In the beginning, the responding tags are divided into M subgroups by using bit masks to reduce the number of timeslots. Then the pilot frame (L_p) is only used with the first subgroup. The other subgroups determine the number of tags without using L_p. the reader creates an additional timeslots at the end of L_p to treat collisions occurred in L_p. the FSAPB algorithm can be grouped into low and high collision probability cases. All tags send their IDs in L_p. then when L_p is over, P_{coll} is computed and compared with the threshold P_{th} [25].

In low collision probability, P_{coll} is lower than P_{th}, binary tree algorithm and additional timeslots L_{add} are used [25].

Fig.19 shows an example of low collision probability case. Tags send their IDs when their counters become zero. Tag 1 and Tag 7 are identified successfully in L_p. tag 2, 3 and 5 collide in L_p, therefore 2 additional timeslots are added to L_p and those tags act binary tree algorithm during L_{add}. Tag 4 and 6 add twice the number of collisions before tag 4 and 6 to the current counter values by the second collision in L_p to selected random binary number 0 or 1. Tag 2 and 5 collide again in L_{add}, they use the binary tree algorithm and other tags increase their counters by 1 [25].

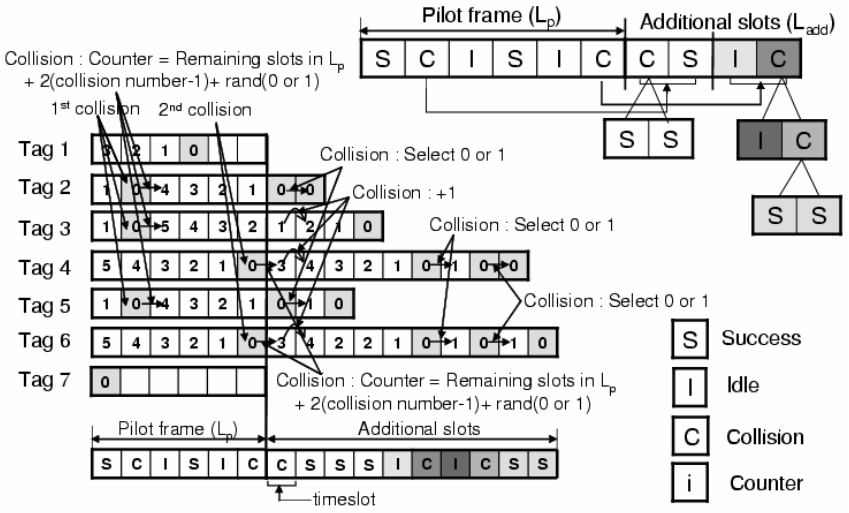


Fig.19: An example of FSAPB algorithm, when Pcoll is lower than Pth [25].

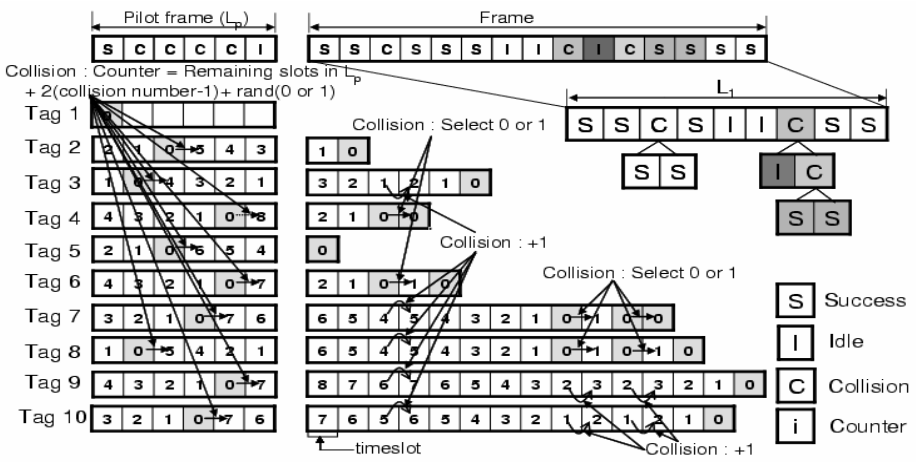


Fig.20: An example of FSAPB algorithm, when Pcoll is larger than Pth [25].

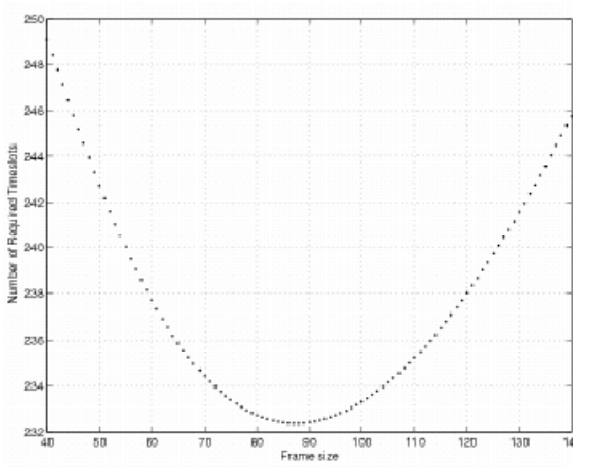


Fig.21: When n=100, the number of required timeslots vs varying frame size [25].

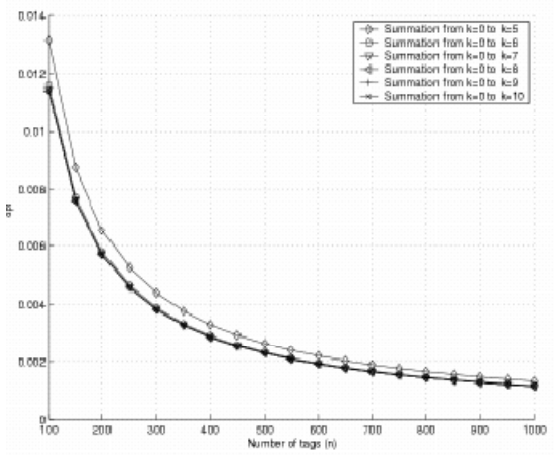


Fig.22: 1/Lopt vs the varying number of tags [25].

In high collision probability, when P_{coll} is greater than P_{th}, most timeslots of L_p are more likely to have collision because number of tags is large. Thus, for tag identification a new frame L_1 is used. The size of L_1 can be set by the minimum n that meet equation (10) [25].

Fig.20 illustrates an example of the high collision probability in FSAPB algorithm. Collided tags in L_p resend their IDs in L_1. Tag 4 and 6 collide in timeslot 3. The other tags in L_1 increase their counters by 1. This process is repeated for other timeslots [25].

The number of tags of the subgroup can be determined after the tags in the first subgroup are identified because bits mask separate tags uniformly. The proper frame size of the current subgroup equals the number of tags identified in the previous subgroup multiplies by a constant (γ).

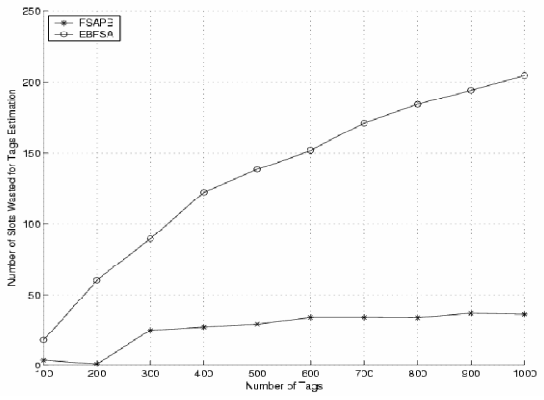


Fig.23: The number of slots wasted for tag estimation with the varying number of tags [25]

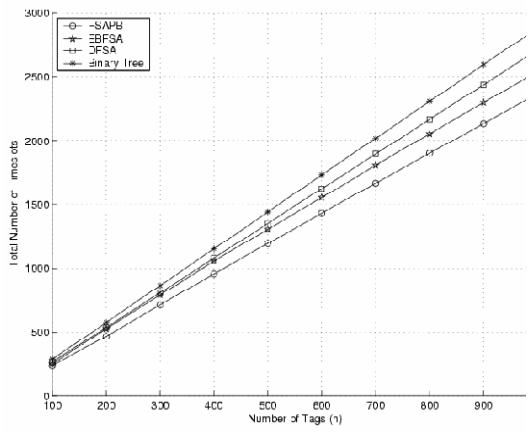


Fig.24: The number of slots used to identify tags with the varying number of tags [25]

When a proper frame size is determined, binary tree algorithm is applied [25].

In DFSA algorithm, the optimal frame size equals the number of tags. However, the optimal frame size in FSAPB algorithm does not equal the number of tags (see fig.21). Thus, the following equation is used to compute the new optimal frame size L_{opt} [25].

$$T = \sum_{k=0}^{n} nC_k \left(\frac{1}{L_{opt}}\right)^k \left(1 - \frac{1}{L_{opt}}\right)^{n-k} L_{opt}(\alpha_k + 1) \quad (11)$$

Where T, n, and k are the total number of timeslots, the number of tags and the number of tags transmitted in a timeslot, respectively. In addition a_k denotes the average number of timeslots used by binary tree when collision occurs. The optimal frame size can be determined by differentiating equation (11). Despite the difficulty of finding the differentiating of equation (11) for the various n, L_{opt} can be determined from fig.22 as 0.88n [25].

Fig.23 shows the number of timeslots wasted for tag estimation for different tags number. It can be seen that FSAPB uses less timeslots than EBFSA for tags estimation. When the number of tags (n < 200), P_{coll} *in* L_p is lower than the threshold. Thus, a reader uses L_{add}, which performs well by binary tree algorithm because of the small number of tags in L_p. However, when the number of tags varies from 300 to1000, L_1 is used for tag identification rather than L_{add} due to a large collision in L_p. Thus, FSAPB does not use more timeslots than L_p. However, EBFSA needs many timeslots as the number of tags is large. When n=1000, EBFSA needs 5.5 times as many timeslots as FSAPB [25].

By comparing FSAPB with ALOHA based and binary tree algorithms, Fig.24 shows the number of timeslots used to identify tags for different tags number. For small number of tags, most of the algorithms have nearly the same performance. However, when the number of tags increases, FSAPB achieves best performance than the other algorithms. With 1000 tags, EBFSA, DFSA and binary tree use more slots about 7.3%, 14.7% and 21.9% ,respectively, than FSAPB [25].

5 Conclusion

In the case of identifying many tags simultaneously using shared communication channel, collision occurs. In this paper, a series of tag anti-collision algorithms in RFID system is surveyed and classified. Many algorithms of ALOHA based methods and tree based methods are described and evaluated. Algorithms which combining the characteristics of two algorithms from the previous methods have the main focus in this paper. These algorithms are called tree-ALOHA method.

QT and TSA show the best performance respectively in tree based method and ALOHA based method. However, tree-ALOHA method shows better performance because of two reasons. Firstly, using tree based method gives 100% identification of all tags by which the throughput is maximized and tag starvation problem in ALOHA method is solved. The second reason is that ALOHA based method shortens the identifying time by which the long time delay in tree based method is solved. Therefore, combining ALOHA method with tree method enhances the performance. For example, as shown in Fig 4.15 DFSA (ALOHA based method) and binary tree (tree based method) use more slots about 14.7% and 21.9%, respectively, in the identification process with 1000 tags than FSAPB (tree-ALOHA based method).

Obviously, the main differences between algorithms in tree-ALOHA method are the way of using the tags IDs (tree technique) in identifying the tags and also adjusting the optimal frame size (ALOHA technique). In further research, it remains challenging to improve these two techniques. For example, in future, using the signal properties such

as signal strength in algorithms may enhance the performance.

References:

[1] L. A. Burdet, "RFID Multiple Access Methods," ETH Zurich 2004.

[2] B. Glover and H. Bhatt, *RFID essentials*. Farnham: O'Reilly, 2006.

[3] M. Bhuptani and S. Moradpour, *RFID field guide : deploying radio frequency identification systems*. Upper Saddle River, NJ: Sun Microsystems/Prentice Hall PTR, 2005.

[4] N. Raza, V. Bradshaw, and M. Hague, "Applications of RFID technology," presented at RFID Technology (Ref. No. 1999/123), IEE Colloquium on, 1999.

[5] L. Biao, H. Ai qun, and Q. Zhong yuan, "Trends and Brief Comments on Anti-collision Techniques in Radio Frequency Identification System," in *6th International Conference on ITS Telecommunications Proceedings*, 2006.

[6] D. H. Shih, P. L. Sun, D. C. Yen, and S. M. Huang, "Taxonomy and survey of RFID anti-collision protocols," *Computer Communications* vol. 29, pp. 2150-2166, 2006.

[7] L. Liu and S. Lai, "ALOHA-Based Anti-Collision Algorithms Used in RFID System," in *Wireless Communications, Networking and Mobile Computing*. China 2006

[8] J. R. Cha and J. H. Kim, "Novel anti-collision algorithms for fast object identification in RFID system," presented at 11th International Conference on Parallel and Distributed Systems Workshops, Fukuoka, Japan, 2005.

[9] C. Lee, H. Cho, and S. W. Kim, "An Adaptive RFID Anti-Collision Algorithm Based on Dynamic Framed ALOHA," *IEICE TRANS. COMMUN*, vol. VOL.E91–B, 2008.

[10] H. Cho, W. Lee, and Y. Baek, "LDFSA: A Learning-Based Dynamic Framed Slotted ALOHA for Collision Arbitration in Active RFID Systems," *Advances in Grid and Pervasive Computing*, pp. 655-665, 2007.

[11] J. Myung, W. Lee, J. Srivastava, and T.K. Shih, "Tag-Splitting: adaptive collision arbitration protocols for RFID tag identification," *In IEEE Trans. on Parallel and Distributed Systems,*, vol. 18 (6), pp. 763-775, Jun. 2007.

[12] K. Finkenzeller, *RFID handbook : fundamentals and applications in contactless smart cards and identification*, 2nd ed. Chichester, England ; Hoboken, N.J.: Wiley, 2003.

[13] L. Liu, Z. Xie, J. Xi, and S. Lai, "An improved anti-collision algorithm in RFID system," presented at 2nd International Conference on Mobile Technology, Applications and Systems, Guangzhou, China, 2005.

[14] J. Myung and W. Lee, "An Adaptive Memoryless Tag Anti-Collision Protocol for RFID Networks," *IEEE ICC*, 2005.

[15] J. R. Cha and J. H. Kim, "Dynamic Framed Slotted ALOHA Algorithms using Fast Tag Estimation Method for FRID system," presented at *3rd IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, United States, 2005.

[16] H. Vogt, "Efficient Object Identification with Passive RFID Tags," in *Pervasive Computing : First International Conference*, vol. Volume 2414/2002. Switzerland: Springer Berlin / Heidelberg, 2002, pp. 98.

[17] X. Wang, G. Xie, Z. Qi, and L. Ming, "Estimation Algorithm of Identifying Object Number in Passive RFID System," presented at Digital Media and its Application in Museum & Heritages, 2007.

[18] S. R. Lee, S. D. Joo, and C. W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," presented at Second Annual International Conference on Mobile and Ubiquitous Systems -Networking and Services, San Diego, CA, United States, 2005.

[19] M. A. Bonuccelli, F. Lonetti, and F. Martelli, "Tree Slotted Aloha: a New Protocol for Tag Identification in RFID Networks," presented at International Symposium on a World of Wireless, Mobile and Multimedia Networks, Buffalo-Niagara Falls, NY, United States, 2006.

[20] T. W. Hwang, B. G. Lee, Y. S. Kim, D. Y. Suh , and J. S. Kim, "Improved Anti-collision Scheme for High Speed Identification in RFID System," presented at First International Conference Innovative Computing, Information and Control, 2006.

[21] J. H. Choi, D. Lee, H. Jeon, J. Cha, and H. Lee, "Enhanced Binary Search with Time-Divided Responses for Efficient RFID Tag Anti-Collision," in *ICC '07. IEEE International Conference Communications*. Glasgow, 2007.

[22] T. P. Wang, "Enhanced binary search with cut-through operation for anti-collision in RFID systems," *IEEE Communications Letters*, vol. 10, pp. 236-238, 2006.

[23] J. Myung and W. Lee, "Adaptive Binary Splitting: A RFID Tag Collision Arbitration Protocol for Tag Identification," *Journal of Mobile Networks and Applications*, vol. 11, pp. 711-722, 2006.

[24] J. H. Choi, D. Lee, and H. Lee, "Bi-slotted tree based anti-collision protocols for fast tag identification in RFID systems," *IEEE*

Communications Letters vol. 10, pp. 861-863, 2006.

[25] J. Eom and T. J. Lee, "Framed-slotted ALOHA with estimation by pilot frame and identification by binary selection for RFID anti-collision," in *ISCIT '07. International Symposium on Communications and Information Technologies*, 2007.

[26] J. Park, M. Y. Chung, and T. J. Lee, "Identification of RFID Tags in Framed-Slotted ALOHA with Robust Estimation and Binary Selection " *Communications Letters, IEEE*, vol. 11, pp. 452-454, 2007.

[27] J. D. Shin, S. S. Yeo, T. H. Kim, and S. K. Kim, "Hybrid Tag Anti-collision Algorithms in RFID Systems," in *Computational Science – ICCS 2007*, 2007, pp. 693-700.

[28] B. Zhen, M. Kobayashi, and M. Shimizu, "to read transmitter-only RFID tags with confidence," presented at IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Barcelona, Spain, 2004.

[29] Y. Kawakita and J. Mitsugi, "Anti-collision performance of Gen2 Air Protocol in Random Error Communication Link," Phoenix, AZ, United States, 2005.

[30] Z. Bin, M. Kobayashi, and M. Shimizu, "Framed ALOHA for Multiple RFID Objects Identification," in *IEICE*, vol. E88-B: Oxford University Press, Bunkyo,Tokyo, 113-0023, Japan, 2005, pp. 991-999.

[31] C. Floerkemeier and M. Wille, "Comparison of transmission schemes for framed ALOHA based RFID protocols," presented at IEEE, Symposium on Applications and the Internet Phoenix, AZ, United States, 2005.

[32] C. Floerkemeier, "Transmission control scheme for fast RFID object identification," presented at Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, Pisa, Italy, 2006.

[33] H. Vogt, "Multiple object identification with passive RFID tags," presented at IEEE International Conference Systems, Man and Cybernetics 2002.

[34] Q. Peng, M. Zhang, and W. Wu, "Variant Enhanced Dynamic Frame Slotted ALOHA Algorithm for Fast Object Identification in RFID System," presented at IEEE International Workshop on Anti-counterfeiting, Security, Identification, 2007.

[35] J. Myung, W. Lee, and J. Srivastava, "Adaptive binary splitting for efficient RFID tag anti-collision," vol. 10, pp. 144-146, 2006.

[36] M. A. Bonuccelli, F. Lonetti, and F. Martelli, "Exploiting id knowledge for tag identification in rfid networks," presented at International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, Proceedings of the 4th ACM workshop on Performance evaluation of wireless ad hoc, sensor,and ubiquitous networks, Greece 2007.

[37] K. W. Chiang , C. Hua , and T. S. P. Yum, "Prefix-Randomized Query-Tree Protocol for RFID Systems," presented at IEEE International Conference Communications ICC Istanbul 2006.

[38] J. L. Massey, "Collision-Resolution Algorithms and Random-Access Communications," California Univ., Los Angeles. School of Engineering and Applied Science, United States 1980.